

Modeling of software process families with automated generation of variants

Andrea Delgado¹ and Daniel Calegari²
Instituto de Computación
Universidad de la República
11300 Montevideo, Uruguay
¹adelgado@fing.edu.uy, ²dcalegar@fing.edu.uy

Félix García
Escuela Superior de Informática
Universidad de Castilla-La Mancha
13071 Ciudad Real, España
felix.garcia@uclm.es

Abstract—Modeling variability in software systems or processes promotes reuse of core assets. In particular, variability in software development processes allows customization of activities, artifacts, roles and other elements to specific projects, in what is called a process variant. Each process variant is derived from a base common process, called process tailoring, which is usually a tedious and error prone manual process. In the last years, there has been a growing interest in supporting the process variability approach, being v-SPEM a proposal that integrates and advanced view of variability into the SPEM standard. In this paper we present an extension of v-SPEM support for final users, with automated generation of variants, to help during process tailoring. The use of automatic mechanisms reduces errors and simplifies the tailoring process by hiding the details to the end users.

Keywords: software development processes, SPEM, process variability, Software Process Lines, Model Driven Engineering.

I. INTRODUCTION

A Software Process is a specific type of business process that focuses on the software development domain, as other domains have their own typical processes such as banking, health, or education. The Software Process Engineering Meta-modeling language (SPEM 2.0 [1]), is commonly used as a domain specific modeling language in this context.

Software processes may accept variants based on specific requirements of an organization or project, leading to the definition of a process family. A Software Process Line (SPrL) [2], [3] aims to provide the techniques and mechanisms to model the common (known as *base process*) and variable parts (known as *variation points*) of a process family, as well as to support the customization of each process variant (known as *process tailoring*). The advantages of using the SPrL paradigm to model software process families are well-known, but although there are many proposals and some tools to support them [4], the tailoring process is in general carried out by final users with the aid of technical users, following a not well defined, manual, error prone and complex process. What is more, existing tools such as Eclipse Process Framework (EPF) Composer¹ which implements SPEM, are mostly targeted to process modelers with process families modeling skills, not to final users.

With the aim of improving the automation of SPrL by reducing technical complexity to final users, the Model-Driven Engineering (MDE) [5] paradigm can be considered as a very promising approach. Based on this, we identify the following research question for our work: *How the automated tailoring of specific variants from a process family can be supported using the MDE approach?*

In this paper, we improve the v-SPEM [3] approach, a proposal of our own that integrates an advanced view of variability into the SPEM standard. The base approach does not formalize the variability process, its tool support is not user-friendly (vEPF²) for non-technical users, and the tailored process is hard-coded. In this context, we extend v-SPEM by defining a process composed by roles, activities and tool support for MDE-based process tailoring. Tailoring is performed by means of an ATL [6] model transformation, based on user's selection. The main contributions of our work are: i) a web tool targeted for final users in which a process family model can be imported and tailored to generate variants; ii) guidance on how to perform such tailoring; and iii) a set of ATL transformations which automatically generate the selected variant.

The rest of this paper is structured as follows. In Section II we briefly present the SPEM standard and v-SPEM extension. In Section III we present related work regarding existing process variability proposals. In Section IV we describe our proposal to support process tailoring within v-SPEM, including transformations for the automated generation of variants, and in Section V we provide an example of application. Finally in Section VI we present conclusions and future work.

II. SPEM AND v-SPEM

The SPEM [1] standard defines specific elements for software process modeling, such as: tasks, activities, roles, work product, and processes. It also provides a way for modeling variability, defining several variability types between two related elements, such as: "replaces" which states that the origin element substitutes the target element under some assumptions, and "extend" which states that the origin element expands the target element definition, probably by overriding attributes and associations.

DOI reference number: 10.18293/SEKE2018-019

¹EPF Composer, <https://www.eclipse.org/epf/>

²v-EPF, https://alarcos.esi.uclm.es/vepf/variant_rich_process_paradigm/vepf.html













	Activity	WorkProductUse	RoleUse	TaskUse
Base Element				
VarPoint				
Variant				

Fig. 1. Main v-SPEM concepts [3]

However, it has some limitations from the perspective of a SP_rL, e.g.: it is not clear how common and variable parts interact, and it is difficult to visualize the base process with the common elements [7]. In this context, the variability extension for SPEM (v-SPEM) [3], [7] provides means for a direct specification of the variability in a process. Variability is modeled in two ways: single (or on-point) variations for expressing variability with respect to specific elements of the process model (e.g.: tasks), and crosscutting variations for expressing variability with respect to several elements at once. Both kinds allow full modeling of software process families. In this work we focus on single variations.

A process engineer defines the variation points over the common process, specifying which variants can occupy these points (i.e.: a "replaces" relation), and during process tailoring, each variation point is substituted with exactly one variant. v-SPEM extends the SPEM metamodel and defines a new notation to represent variants and variation points for SPEM elements, as shown in Figure 1, in which a base SPEM element, e.g.: *Activity*, can be defined as a variation point (*VPActivity*) or as a variant (*VActivity*).

It also provides tool support as an extension of the EPF Composer, as mentioned, the v-EPF The tool allows modeling a process family and the Java-based generation of process variants. However, it is devised exclusively for technical users, and hard-codes the generation of variants.

III. RELATED WORK

Some proposals are also based on SPEM 2.0 models. SmartySPEM [8] adds variability over SPEM by adding stereotypes to the metamodel. It provides guidelines for identifying and representing process variability, and process tailoring by variability resolution. Tool support is using any UML editor, which are as EPF, targeted to technical users. It was defined after v-SPEM and provides similar elements, and no generation of variants. In CASPER [9] the authors define two input models: the base process with variability, and a context model for a specific project, specifying things such as: project size, schedule, team size, among others. The approach provides automated support for process tailoring based on a ATL [6] model transformation, which takes the context model for the inference of variants. Then, variants are deduced from the values entered by the final user, leading to a variant they

generate based on previous knowledge. In this approach, the final user is not able to choose between variants for the variation points.

Other works are based on feature models (FMs), which is the common approach in software product lines. In [10] the authors use FMs for expressing variability with respect to roles, tasks and work products. One drawback is that features and relations of different types cannot be distinguished. Also, the control flow of the process is lost. In [11] the authors use a SPEM model for identifying variation points, and an Orthogonal Variability Model associates each variation point with the defined variants (as with FMs).

The Common Variability Language (CVL)³ [12] is an approach for language independent variability modeling. Besides that automated generation of variants is supported (but not provided), models are hard to specify and maintain.

As a summary, all approaches provide support for variability almost over the same type of elements. Most of them also provide supporting tools, but not all are available. Also, all approaches provide guides for process tailoring. Besides CASPER, the generation of process variants in the other approaches is carried out manually. Unlike CASPER, we let the user choose each specific variant for each variation point.

IV. v-SPEM TAILORING SUPPORT

In what follows we describe the roles, activities, process tailoring and tool support we propose for v-SPEM.

A. Roles

When managing process families, there are different stakeholders involved, who participate in different activities and are interested in different artifacts or work products, as in the software development process itself. We have defined two main roles within our proposal, the process engineer (technical user) and the final user (non-technical user).

The Process engineer role, is in charge of modeling the process family, thus it needs advanced software engineering skills such as knowing about SPEM, v-SPEM, SP_rL and process families, being familiar with process modeling and using tools such as v-EPF for doing it.

The Final user can be a software project manager who has knowledge of software engineering and software development processes but, not necessarily about SP_rL and process families. However, this user is the one who needs to tailor base processes into a specific variant for each project he will lead. Thus, for this kind of user, it is very valuable to have extra support to carry out the tailoring process.

Based on these definitions we defined the main activities each one must perform within the tailoring process with the Use Cases shown in Figure 2. As it can be seen, the process engineer is mostly involved with the technical activities to define the base process, variability points and possible variants to occupy them. The final user is the focus of the tailoring process, for which we extended the v-SPEM existing support. Also, we add the possibility of working with a central

³CVL. <http://www.omgwiki.org/variability/doku.php>

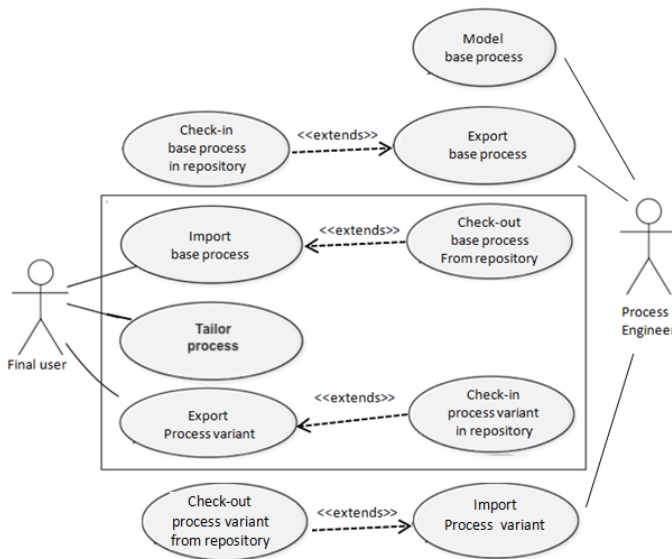


Fig. 2. Use cases for the tailoring process

repository for models, both the base process and the generated variants, such as GitHub⁴. The advantage of this is that the organization can manage their models in a centralized and consistent way, promoting the reuse of elements.

B. Process tailoring

The result of the process tailoring is a process variant, defining a complete software development process, with no variability, for the specific project we are customizing it to. The process tailoring is part of the more general managing process variability process, that is shown in Figure 3 modeled with BPMN 2.0 as a business process.

As depicted in Figure 3, *Model base process* and *Tailor process* are defined as sub-processes, meaning that several tasks need to be carried out in order to obtain the base process and then, from it, a process variant. We do not explicit the *Model base process* sub-process since it is not the focus of this paper, but we detail the *Tailor process* sub-process.

After obtaining the base process of the family (locally or from the central repository) the final user has to derive the specific variant from it, tailoring the base process. For doing this, the final user has to select each variation point and, from the possible variants, select the one that best fits the project at hand. After this, an automated activity is performed to check the restrictions associated with the selected variation point and corresponding variant. An example of this can be if there was a previous variation point that should have been filled with a specific variant say A, to be able to select the variant B in the following variation point. If the restrictions hold the final user can continue selecting variation points and variants for it, until there are no more variations point. When the restrictions do not hold, the variant cannot be selected.

We enclose the import-export tasks in groups, to explicitly show the interactions between the process engineer and the final user via the base process and process variant generated. This supports the use of a centralized repository, where all base process and all generated variants can be stored and reused within the organization. These tasks could have not been shown in the process, since their corresponding work products, i.e.: the process models, can also be seen as input and output data from other activities. We explicitly model them as tasks, since they show user tasks that the defined roles have to carry out within the process.

Finally, after process tailoring is manually defined, we execute an ATL [6] model transformations, automatically generating the corresponding process variant. This transformation takes as an input the base process and the configuration model with the information of the variants that the user selected to occupy each variation point. For now, this variant has to be imported in the EPF so the process engineer is able to publish it as a web site using the facilities provided by the framework. However, we are working on integrating these facilities in the web site, so the generated process variant can be also published by the final user.

C. Automated generation of variants

The automated generation of variants provides support for a cleaner and repeatable tailoring process, with knowledge reuse and less human errors. The ATL transformation takes as input the base process which was modeled by the process engineer, and the configuration model containing the information of the variants selection made by the final user, when tailoring the process. The transformation contains the knowledge regarding which key elements of the variability modeling extension in the vSPeM metamodel can be mapped to which key elements of the SPeM metamodel, used to specify the generated process variant, based on the information in the configuration model.

Since the base process of the process family also contains elements that are common to all process variants of the family, these elements with no variability have to be maintained in the transformation, as they must be included in all process variants. For this reason, we have separated the generation in two main blocks: i) elements with no variability that have to be copied as they are into the output process variant, and ii) variation points (i.e.: elements with variability) whose selected variant is defined in the configuration model, thus replacing the variation point in the base process with the selected variant.

In Listing 1 we present an excerpt of an adaptation rule to generate a variant from an `Activity` variation point. The transformation takes a model conforming to the v-SPeM metamodel (`Vuma.ecore`) and the configuration model (`Configuration.ecore`) and produces another v-SPeM model (the process variant). The rule takes an activity variation point (`vpActivity`), selects the variant (by id in the configuration model using and auxiliary function) that must be used (`VarActivity`), and generates as output an activity with the information provided by the selected variant.

⁴GitHub. <https://github.com/open-source>

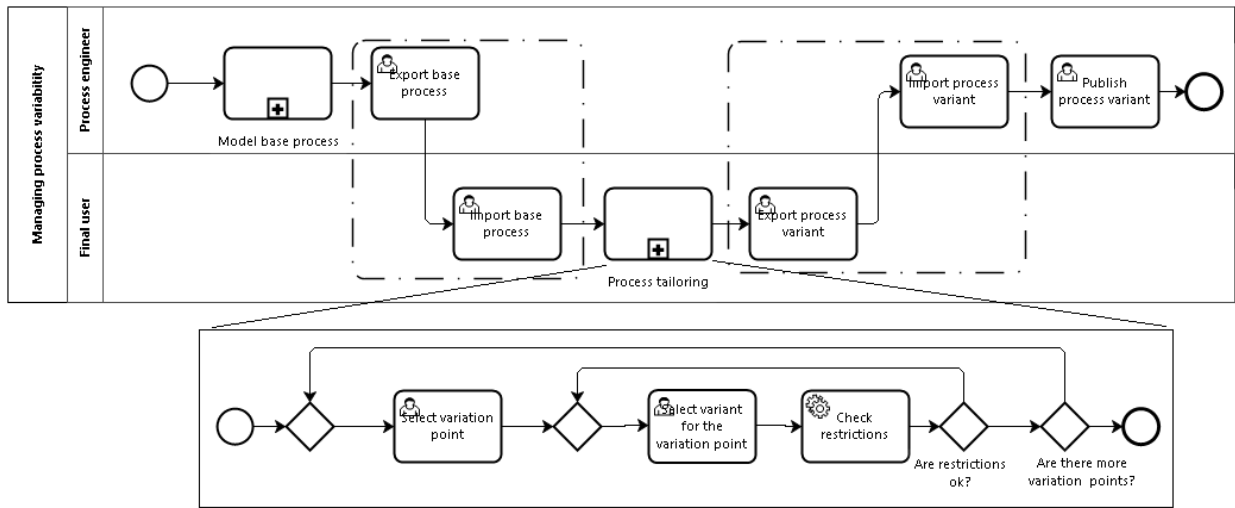


Fig. 3. Managing process variability process

Listing 1. Example of a variant generation rule

```

-- @path MM=/Vuma.ecore
-- @path MM1=/Configuration.ecore
-- @path MM2=/Vuma.ecore

module vSpemYconf;
create OUT: MM2 from IN: MM, IN1: MM1;
.....

//rule to adapt an Activity
rule AdaptActivity {
  from
    input_name: MM!vpActivity (
      input_name->vpProcessElementVa()
    )
  using {
    variant: MM!VarActivity = MM!VarActivity
      -> allInstances() -> any(vt | vt.guid
        = thisModule ->
          getSelectedVariantOfVarPoint
            (input_name.guid).id);
  }
  to
    output_name: MM2!Activity (
      name <- 'AdaptedActivity:' +
        variant.name,
      guid <- input_name.guid,
      presentation <- variant.presentation
    )
}

```

D. Tool support

As mentioned before, we use as basis the v-EPF tool that is an extension of EPF to support the v-SPEM proposal. It adds a new folder named Process Lines in which Capability pattern and Delivery process with variability are defined. SPRLs in this folder are structured as shown in Figure 4. Folders *Var Points* and *Variants* contain the variation points and variants defined, also holding the dependence relations

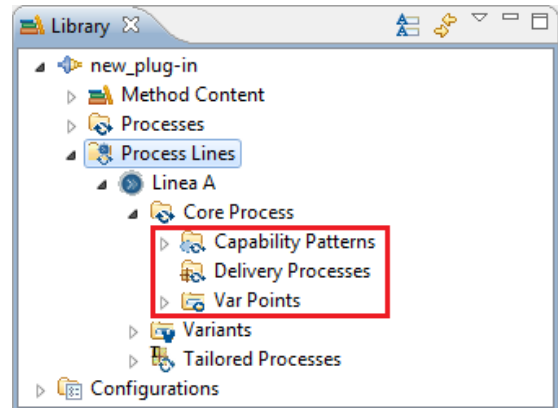


Fig. 4. v-EPF tool supporting the vSPEM extension

between variants and variation points. In the Tailored process folder, the process variants that can be derived are specified. Although v-EPF already provides support for the process tailoring, it is aimed at technical users with broadly knowledge of EPF and SPEM. Also, process variability does not allow nested variability, and publishing the process in the web is not allowed for processes defined below the process line folder.

Based on the definitions we have presented in previous sections, we also extended the v-EPF tool to support process tailoring with focus on the non-technical final user. For this, we have developed a web application shown in Figure 5.

The web application we have implemented can be used by any final user in the organization and works together with the central repository and the v-EPF used by the process engineer. Figure 5 shows at the top the three main activities (tasks and sub-process) to be performed by the final user as defined in the BPMN 2.0 process shown in Figure 3: (1) import the base process from the repository, (2) perform process tailoring by resolving every variation point (once it is done, the application runs the model transformation), and (3) export process variant.

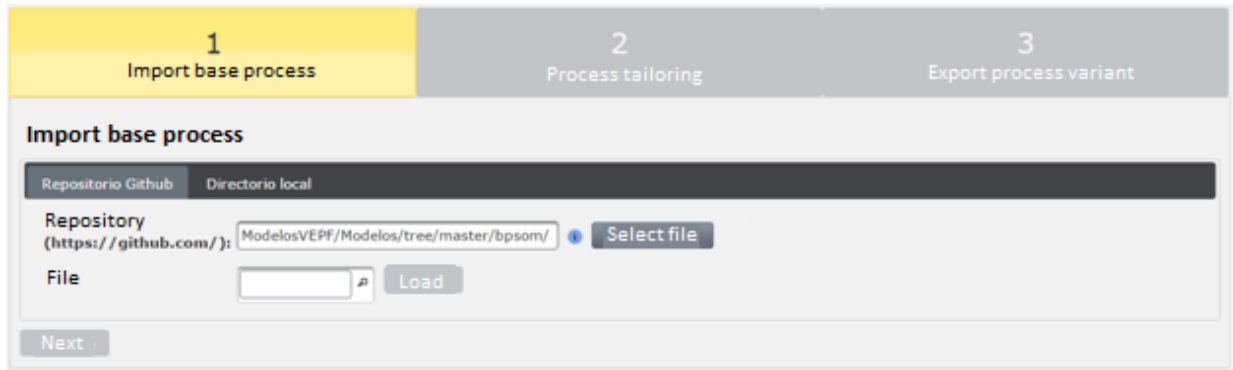


Fig. 5. Screen-shot of the v-SPeM web tool

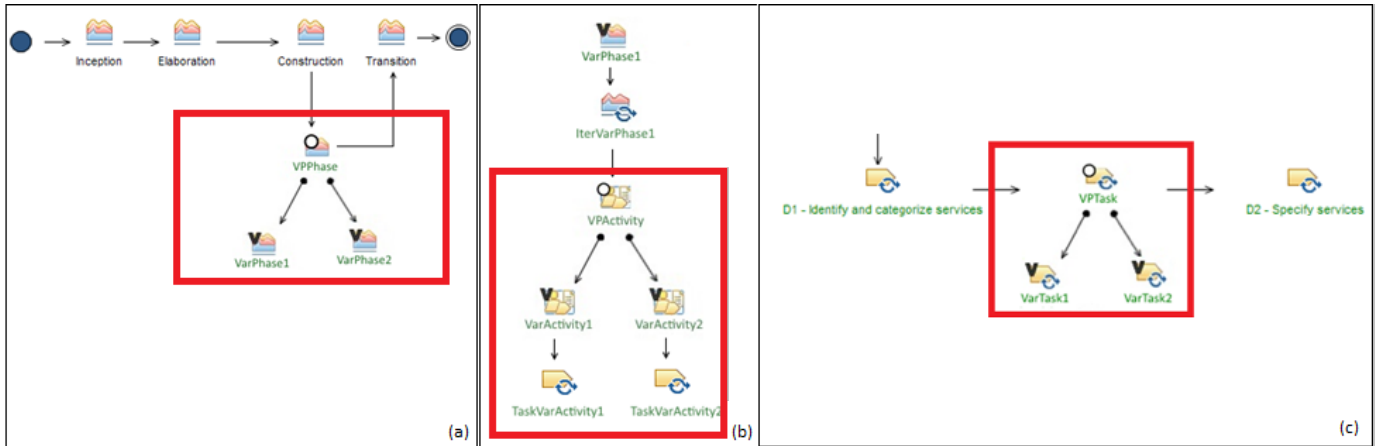


Fig. 6. BPSOM variability with v-SPeM: (a) phase, (b) activity, (c) task

V. EXAMPLE OF APPLICATION

In this section we present an example of application of our proposal based on the BPSOM⁵ software development process we have defined for implementing service-oriented applications from business processes [13]. The process was specified using EPF. It is based on the Unified Process and customized with specific roles, disciplines, activities, work products and a delivery process consisting of four phases.

The process engineer uses v-EPF, under the process line category, to model the base process adding variability in the elements that v-SPeM provides support for: phases, iterations, activities, tasks and roles, as shown in Figure 6. In (a) a new phase between the construction and transaction phases was added, with variants that also have variability for one activity and for one iteration (b); in (c) variability for a task was added, which also has role variability in the corresponding variants. Then, the process engineer exported the base process with variability to a GitHub repository.

The web application allows a final user to perform process tailoring in three steps. First, the base process is imported from the repository. Second, process tailoring is performed as shown in Figure 7. The process is graphically depicted, in this

case: the phases of the BPSOM base process with the phase variation point we added. The variation point has a link that, when pressed, shows the defined variants that can be selected to occupy its place. After selecting the `varPhase1` variant for the `VPPHase` variation point, the variant corresponding to the nested `VPAActivity` variation point must be also defined. In this case, the `VarActivity2` variant is selected, which includes the task `TaskVar2`. This variant also requires to select the role that will perform the `TaskVar2` task, thus the `VarRole1` is selected. Third, the generated variant can be exported to EPF and published as a web site, which facilitates its use as the software development process for the selected project within the organization.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we presented a proposal that provides support for process tailoring in SPeL based on v-SPeM. Although the approach already provides support for managing variability in SPeL, it does not provide guidance for final users (there were no explicit process) and its tool support for carrying out the process tailoring was hard-coded and within a technical user environment. With this motivation, we specified the activities to be performed for the defined roles and extended the tool support with a new web application focused on process

⁵<http://alarcos.esi.uclm.es/minerva/bpsom/published/>

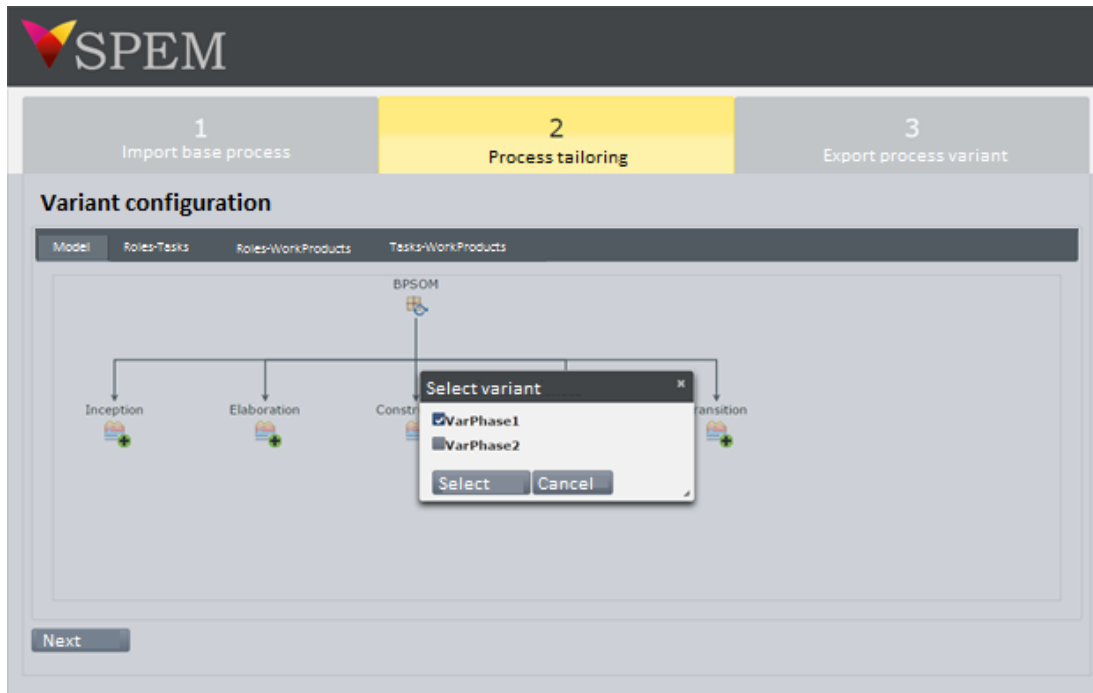


Fig. 7. Web process tailoring: selection of a variant for a variation point

tailoring for final users. The tool provides guides and support for each activity in the tailoring process. Besides it seems that tool support was improved, since the web application is friendlier for non-technical users than the Eclipse environment, and also technical complexity is completely hidden for them, we require further empirical studies with final users in order to conclude. In particular, we expect to evaluate and improve the approach by carry out a case study in a real organization.

The tool also provides organizations with a central repository for SPRL, laying the foundation for a systematic governance of their processes. Moreover, the automated generation of variants provides support for a clearer and less error prone process with respect to the traditional manual tailoring process.

We expect to improve the tool, as for example, publishing generated process variants directly from the web application without the round trip to v-EPF. Moreover, we expect to improve our work for supporting crosscutting variations, as defined in v-SPEM. Finally, although the paper focuses on software processes, the generation of variants approach is quite generic. In fact, we are currently working on applying the approach in the context of BPMN 2.0 business processes [14].

ACKNOWLEDGEMENT

This work was partially funded by Comisión Sectorial de Investigación Científica (CSIC), Uruguay. We would like to thank undergraduate students Fabiana Roldán and Marcela Viera, and CSIC grantees Emiliano Alonzo and Martín Prino who worked with the tool and transformation.

REFERENCES

[1] OMG, “Software and Systems Process Engineering Metamodel (SPEM) v2.0,” Object Management Group (OMG), Tech. Rep., 2008.

[2] D. Rombach, “Integrated software process and product lines,” in *Unifying the Software Process Spectrum: Intl. SW Process Works. 2005, Revised Sel. Papers.* Springer, 2006, pp. 83–90.

[3] T. Martínez-Ruiz, F. García, and M. Piattini, “Towards a spem v2.0 extension to define process lines variability mechanisms,” in *SW Eng. Research, Mgmt and Applications (SERA)*. Springer, 2008, pp. 115–130.

[4] T. Martínez-Ruiz, J. Münch, F. García, and M. Piattini, “Requirements and constructors for tailoring software processes: a systematic literature review,” *Software Quality Journal*, vol. 20, no. 1, pp. 229–260, 2012.

[5] S. Kent, “Model driven engineering,” in *Integrated Formal Methods, Third Intl. Conf., IFM 2002, Proceedings*, ser. Lecture Notes in Computer Science, vol. 2335. Springer, 2002, pp. 286–298.

[6] F. Jouault, F. Allilaire, J. Bezivin, and I. Kurtev, “Atl: A model transformation tool,” *Science of Computer Programming*, vol. 72, no. 1–2, pp. 31 – 39, 2008.

[7] T. Martínez-Ruiz, F. García, M. Piattini, and J. Munch, “Modelling software process variability: an empirical study,” *IET Software*, vol. 5, no. 2, pp. 172–187, 2011.

[8] E. Oliveira, M. Pazin, I. Gimenes, U. Kulesza, and F. Aleixo, *SMartySPEM: A SPEM-Based Approach for Variability Management in Software Process Lines*. Springer, 2013, pp. 169–183.

[9] J. Hurtado and M. Bastarrica, “Building software process lines with casper,” in *Intl. Conf. on Software and System Process.* IEEE Press, 2012, pp. 170–179.

[10] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, “Feature-oriented domain analysis (foda) feasibility study,” Carnegie Mellon University, Tech. Rep. CMU/SEI-90-TR-021, 1990.

[11] K. Pohl, G. Böckle, and F. van der Linden, *Software Product Line Engineering*. Springer, 2005.

[12] E. Rouille, B. Combemale, O. Barais, D. Touzet, and J. Jezequel, “Leveraging cvl to manage variability in software process lines,” in *2012 19th Asia-Pacific Software Engineering Conference*, 2012, pp. 148–157.

[13] A. Delgado, F. Ruiz, I. García, and M. Piattini, “Business process so methodology (BPSOM) with service generation in soaml,” in *Advanced Inf. Systems Eng. - 23rd Intl. Conf., CAiSE 2011*, 2011, pp. 672–680.

[14] A. Delgado and D. Calegari, “BPMN 2.0 based modeling and customization of variants in business process families,” in *XLIII Latin American Computer Conference, CLEI 2017*. IEEE, 2017, pp. 1–9.